AMENDMENTS TO THE CLAIMS

1. (Currently Amended) A method, comprising:

processing a type-safe platform-independent firmware component during a pre-boot phase of a computer platform, the type-safe platform independent firmware component is processed using an intermediate language (IL) interpreter for IL; and

processing the type-safe platform-independent firmware component during an operating system- (OS)-runtime phase, the type-safe platform independent firmware component is processed using a Just-in-Time (JIT) compiler for the IL .

2. (Currently Amended) The method of claim 1, wherein the type-safe platform-independent firmware component is written in a type-safe intermediate language (IL). ~~and is processed using an interpreter for the IL during the pre-boot phase~~.

3. (Original) The method of claim 1, wherein the type-safe platform-independent firmware component is written in a type-safe intermediate language (IL) and is processed using a Just-in-Time (JIT) compiler for the IL during the pre-boot phase.

4. (Original) The method of claim 3, wherein processing the IL with the JIT compiler generates an executable image, further comprising storing the executable image to be accessible to platform firmware during a subsequent pre-boot phase for the computer platform.

5. (Original) The method of claim 4, wherein the executable image is stored on a platform firmware storage device.

6. (Original) The method of claim 4, wherein the executable image is stored in a portion of a disk drive that is accessible to the platform firmware.

7. (Currently Amended) The method of claim 1, wherein the type-safe platform-independent firmware component is written in a type-safe intermediate language (IL). ~~and is processed using a Just-in-Time (JIT) compiler for the IL during the OS-runtime phase~~.

8. (Original) The method of claim 1, wherein the type-safe platform-independent firmware component is written in an intermediate language in accordance with the Common Language Infrastructure (CLI) standard.

9. (Original) The method of claim 1, further comprising performing a type-safety verification on the type-safe platform-independent firmware component prior to processing it during the pre-boot phase.

10. (Original) The method of claim 1, further comprising performing a type-safety verification on the type-safe platform-independent firmware component prior to processing it during the OS-runtime phase.

11. (Currently Amended) A method comprising:

encoding source code corresponding to a firmware driver into a type-safe intermediate language (IL)-encoded firmware driver; ~~and~~

processing the type-safe IL-encoded firmware driver during a pre-boot phase of a computer platform using an intermediate language (IL) interpreter for IL; and

processing the type-safe IL encoded firmware driver during an operating system (OS) runtime phase using a Just-in-Time (JIT) compiler for the IL.

12. (Original) The method of claim 11, wherein the type-safe IL-encoded driver comprises a PE/COFF (Portable Executable/Common Object File Format) image.

13. (Original) The method of claim 11, wherein the type-safe IL-encoded driver is encoded into IL code defined by the Common Language Infrastructure (CLI) standard.

14. (Original) The method of claim 11, further comprising verifying the type-safe IL-encoded firmware driver for type-safety prior to its processing.

15. (Original) The method of claim 11, wherein the type-safe IL-encoded firmware driver is processed using an IL interpreter.

16. (Original) The method of claim 11, wherein the type-safe IL-encoded firmware driver is processed using a Just-in-Time (JIT) compiler.

17. (Original) The method of claim 11, wherein the type-safe IL-encoded firmware driver is compliant with the Extensible Firmware Interface (EFI) standard.

18. (Original) The method of claim 11, wherein the type-safe IL-encoded firmware driver is accessed during OS-runtime by an operating system user mode.

19. (Original) The method of claim 11, wherein the type-safe IL-encoded firmware driver is accessed during OS-runtime by an operating system kernel mode.

20. (Currently Amended) A machine-readable media to provide instructions, which when executed perform operations comprising:

loading a type-safe processor-neutral firmware module into a pre-boot environment; ~~and~~

loading the type-safe processor-neutral firmware module into an operating system runtime environment; and

processing the type-safe processor-neutral firmware module via a virtual processor in the pre-boot environment.

21. (Original) The machine-readable media of claim 20, wherein the instructions including native instructions corresponding to an interpreter to operate as the virtual processor.

22. (Original) The machine-readable media of claim 20, wherein the instructions including native instructions corresponding to a Just-in-Time (JIT) compiler to operate as the virtual processor.

23. (Original) The machine-readable media of claim 20, wherein the instructions include instructions written in intermediate language (IL) code.

24. (Original) The machine-readable media of claim 23, wherein the IL code is compliant with the Common Language Infrastructure (CLI) standard

25. (Original) The machine-readable media of claim 20, wherein the media comprises a firmware storage device, and the instructions comprise firmware instructions.

26. (Original) The machine-readable media of claim 20, wherein execution of the instructions performs the further operation of publishing an interface via which an operating system (OS) may access the type-safe processor-neutral firmware module during OS runtime operations.

27. (Currently Amended) A computer system, comprising:

a system processor,

memory, coupled to the processor; and

a flash device on which firmware is stored, said firmware including a plurality of firmware drivers written in a type-safe intermediate language and native instructions, the flash device further including ~~comprising~~ a virtual processor to be hosted by the system processor, said virtual processor to process the plurality of firmware drivers during a pre-boot phase for the computer system.

28. (Original) The computer system of claim 27, wherein the type-safe intermediate language is compliant with the Common Language Infrastructure (CLI) standard.

29. (Original) The computer system of claim 28, wherein the virtual processor comprises a CLI-compliant interpreter.

30. (Original) The computer system of claim 28, wherein the virtual processor comprises a CLI-compliant Just-in-Time (JIT) compiler.